**高性能索引**

例：SELECT \* FROM award WHERE nickname = 'css'

没建立索引就全表扫描你需要的数据

在nickname上建立索引,那么mysql只需要扫描一行数据及为我们找到这条nickname='css'的数据

单列索引(主键索引,唯一索引（可以空值）,普通索引)和组合索引.

单列索引：一个索引只包含一个列,一个表可以有多个单列索引.

组合索引：一个组合索引包含两个或两个以上的列

最左前缀规则：(最频繁查询的列放左边)

如果你建立了 组合索引(a\_b\_c) 那么他实际包含的是3个索引 (a,b)(a,c)(a,b,c)

以where ‘a’= ？？ 开头 不可以只查询b和c

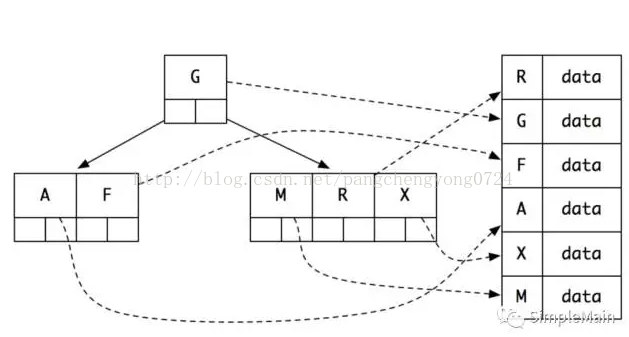
CREATE INDEX IndexName ON `TableName`(`字段名`)

索引（Index）是帮助MySQL高效获取数据的数据结构。

数据本身之外，数据库还维护着一个满足特定查找算法的数据结构，这些数据结构以某种方式指向数据，这样就可以在这些数据结构的基础上实现高级查找算法，这种数据结构就是索引

B-Tree能加快数据的访问速度，因为存储引擎不再需要进行全表扫描来获取数据，数据分布在各个节点之中

通常，数据库的索引一般值B-Tree索引，Mysql中InnoDB引擎与MyISAM引擎具体实现都是采用B+Tree算法。

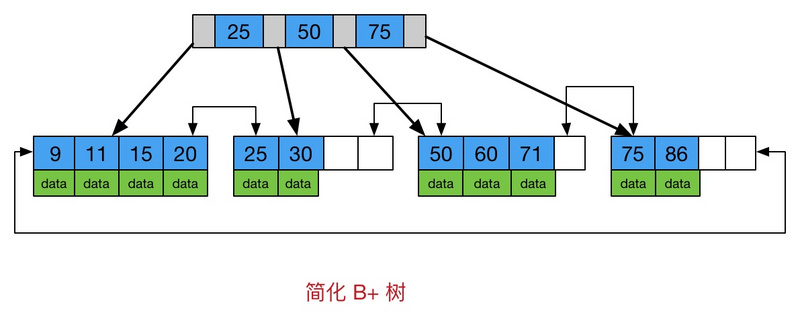


**B树里面，每个结点有这样的特点：**

**不论是叶结点还是非叶结点，都含有子节点指针和一个指向数据的指针**

**带来的问题，每一个数据的指针会带来额外的内存占用，从而减少放入内存的结点数**

mysql是基于磁盘的数据库，索引是以索引文件的形式存在于磁盘中的，索引的查找过程就会涉及到磁盘IO消耗，磁盘IO的消耗相比较于内存IO的消耗要高好几个数量级，所以索引的组织结构要设计得在查找关键字时要尽量减少磁盘IO的次数

B+Tree索引  
B+Tree的结构和B-Tree类似，不过它的非叶子结点只包含子节点指针，不包含指向数据的指针，所有的叶子结点使用链表相连，便于区间查找和遍历

**为什么说B+树比B树更适合实际应用中操作系统的文件索引和数据库索引？**

**B+树的磁盘读写代价更低**

**B+树的内部结点并没有指向数据的指针。可以将更多索引数据载入内存，IO读写次数也就降低了。**

**B+树的查询效率更加稳定**

**由于非终结点并不是最终指向文件内容的结点，而只是叶子结点中关键字的索引。所以任何关键字的查找必须走一条从根结点到叶子结点的路。所有关键字查询的路径长度相同，导致每一个数据的查询效率相当。**

Hash索引

对于每一行记录，存储引擎会对所有的索引列计算一个哈希码，哈希索引将所有哈希码存储在索引中，同时在哈希表中保存指向每个数据行的指针。

哈希索引只包含哈希值和行指针，而不存储字段值。

哈希索引无法用于排序，因为不是根据索引值顺序存储的。

哈希索引不支持部分索引列匹配查找，因为是使用所有索引列来计算哈希值的。 ​

聚簇索引与非聚簇索引

聚簇索引并不是一种单独的索引类型，而是一种数据存储方式。Mysql中InnoDB引擎采用的就是聚簇索引来存储数据。

**索引的优缺点**

**优势**

**类似大学图书馆建书目索引，提高数据检索的效率，降低数据库的IO成本。**

**通过索引列对数据进行排序，降低数据排序的成本，降低了CPU的消耗。​**

**缺点**

**实际上索引也是一张表，该表保存了主键与索引字段，并指向实体表的记录，所以索引列也是要占用空间的。**

**虽然索引大大提高了查询速度，同时却会降低更新表的速度，如对表进行INSERT、UPDATE和DELETE。因为更新表时，MySQL不仅要保存数据，还要保存一下索引文件每次更新添加了索引列的字段，都会调整因为更新所带来的键值变化后的索引信息。**

**索引的选择性是指索引列中不同值的数目与表中总记录数的比。一个索引的选择性越接近1，这个索引的效率就越高。**

这里举两个例子说明一下。

假如一个表有10万行记录，有一个字段A只有T和F两种值，且每个值的分布概率大约为0.5，这时索引的选择性是2/100000=0.00002，那么对该表的字段A建索引一般不会提高数据库的查询速度。

再假如如果一个表中2000条记录，表索引列有1980个不同的值，表索引列有1980/2000=0.99，可以建立索引。

**需要建立索引：**

1. **主键自动建立主键索引**
2. **频繁作为查询条件的字段应该创建索引(where 后面的语句)**
3. **查询中与其它表关联的字段，外键关系建立索引。**

**A 表关联 B 表： on 后面的连接条件,即A 表查询 B 表的条件。**

**所以 B 表被关联的字段建立索引能大大提高查询效率。**

1. **单值/复合索引的选择问题，在高并发下倾向创建复合索引。**
2. **查询中排序的字段，排序字段若通过索引去访问将大大提高排序速度。**
3. **查询中统计或者分组字段**